verification method, the cloud server is not able to provide an effective integrity proof for the given data unless all the data is complete.

To ensure the integrity of the user data in the cloud server, data auditing has more importance than any other data protection mechanism in the cloud server [17]. In this case it provides the verifier as a trustworthy, direct and intelligence in real time of integrity of user data by requesting a challenge. A user having a highly confidential data has to be audited on a regular interval. Many existing auditing schemes have many properties and potential risks. It has inefficiencies such as processing unwanted small updates, security risk in case of unauthorized auditing request. In this paper, we focus better support in small dynamic updates and provides efficiency and scalability in the cloud server. To perform better the proposed method, use ranked Merkle hash tree (RMHT) and flexible data segmentation. We address a security problem in public verifiability which makes the method more robust and secure. To achieve this some additional authorization process are included in Cloud Server, client and third-party auditor (TPA).

2. Related Work

Elasticity and scalability are the key advantage of cloud while comparing with the traditional systems. Efficiently supporting the dynamic data is the great importance. Protection of dynamic data are studied in past [9], [10], [15], [18]. This paper focus on regular updates on small data, since these updates are done in several cloud applications namely social networks and business transactions [19]. Cloud user splits the big data sets into small blocks and store it in different server locations for privacy preserving, reliability and efficient processing. The main problems in the cloud is data security and privacy [20] and it has the main frequent concern [21]. There are lot of research to enhance data security and privacy in Cloud Service provider (CSP) side [22][23].

Integrity verification in outsourced data has attracted research interest. Jules et al. [24] proposed the first model of proofs of retrievability (POR) but this method is only applied to the static data such as library or archive. Ateniese, et al. [14] proposed a method called ‘provable data possession’ (PDP). This method offer block less verification ie., integrity of the outsourced data can be verified by combining pre computed tags such as homomorphic linear authenticators (HLAs) or homomorphic verifiable tags (HVTs). Shacham et al. [25] proposed an improvement on POR model by including stateless verification. They proposed MAC based verification method and initiated the public verification method based on BLS signature [26]. In their next method, the verification and generation for integrity proof is similar to BLS signature verification. When there is a similar security strength (80-bit security) then BLS signature is shorter than the RSA signature and it denotes a benefits of POR scheme. From the concept of POR and PDP method a new compact POR model is derived.

Ateniese et al. [12] proposed a new model which supports scalability, but this model supports only predefined challenges and partial data updates. Erway et al. [16] proposed a PDP model with skip list and supports full dynamic data updates. But, by default it will not support variable size data block and public auditability. Wang et al. [10] proposed a model that supports full data dynamics and public auditing is done by the third-party auditor (TPA). The public auditing is done by dynamic support. But this model lacks in authorized auditing and fine-grained update. Wang et al. [9] proposed a random masking method that ensures TPA cannot able to infer with the raw data from a group of integrity proof. Ateniese et al. [27] proposed a method of transforming mutual identification protocol to PDP scheme. Zhu et al. [28] proposed a method that several hybrid cloud service providers proves data integrity to their data owner. J. L. Joneston Dhas et al. [29] proposed a framework to secure the sensitive data in public cloud. Curtmola et al. [15] proposed a method that proves the integrity of several replicas with original data.

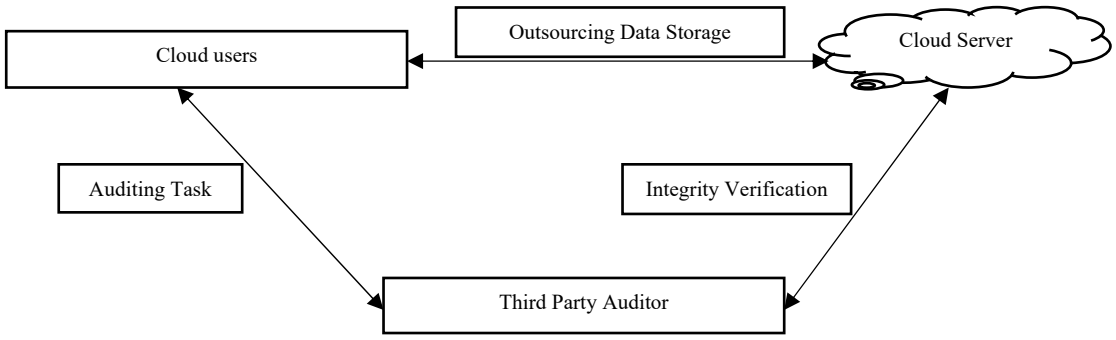

Figure 1: Big Data Security model

3. Problem Analysis

3.1. *Participating parties roles*

Most POR and PDP methods supports the verification of public data. In this method three parties are participated: Client, Cloud Server and TPA. Figure 1 shows the relationship of three parties. For clients the TPA and cloud server are semi-trusted. In the previous old models challenge messages are very simple and any one can send the challenge to the cloud server to get a proof of a certain block. Which enables malicious exploits. First, the malicious party launch a distributed denial-of-service (DDOS) attacks. The party sends multiple messages from different clients that causes an additional overhead in cloud server which makes congestion in the network connections and the quality of services is degraded. Second, by the integrity proof the adversary can get the sensitive information which is returned by cloud server. While challenging the cloud server multiple times the adversary can get the information about the user data or retrieve the statistical information about the status of the cloud service. Even the traditional PDP methods supports the public verifiability it cannot meet 'auditing-as-a-service' security requirement.

3.2. *Operations in verifiable fine-grained dynamic data*

Some few existing public auditing schemes supports full dynamic data [5], [11], [16]. In all this method the operations such as insertion, deletion and modification in fixed-size blocks are discussed. In BLS- signature schemes [10], [16], [17] has 80-bit security and the size of the data blocks is restricted by 160-bit in prime group of order p, all blocks are segmented into the fixed size of 160-bits. This method is unsuitable for variable-sized data blocks and has an advantage in shorter integrity proof. In fact, the existing method supports insertion, deletion or modification in one or many fixed size blocks and it is called as coarse-grained updates.

The coarse-grained update provides the integrity verification methods with minimum scalability and the operations for data updating is more complicated. The data verification update process in [5], [13] cannot able to handle deletion or modification where the size is less than the block. For insert operations there has simple extension and enables insertion in arbitrary size data set. For each insertion a new block is created. When there has a huge number of small updated the storage is wasted hugely. In [5], [13] the data block size is considered as 16k bytes and in each 140-byte message insertion nearly 99 percent new allocated memory is wasted. The user cannot able to reuse the memory until that block has deleted. All these problems are resolved when it supports the fine-grained data update method. According to the observation, usage of fine-grained updates brings additional flexibility and the efficiency is improved. The PDP and POR models are different and their goals are same. The main difference between them is that in POR model the file encoding is done with the error correction and not in PDP model.

4. The Proposed Method

In the proposed method the aim is to support variable size data block, fine grained dynamic data updates and authorized third-party auditing.

4.1. *Overview*

(1) The key generation for client is generated by File Process (FP) and Key Generation (KG) and the data is uploaded to the cloud server. The client stores a RMHT as metadata instead of Merkle Hash Tree (MHT). The TPA will authorize by client by sharing the value of Signature Authentication (SA).
(2) The client fine grained update request is performed by the cloud server by Perform Update (PU) the Verify Update (VU) is run by client to check the updates are done both in data block along with corresponding authenticators which is used in auditing.
(3) TPA verifies integrity for the stored data in the cloud server by Challenge, Verify and Proof.

The following chapter describe the proposed model in detail

(1) Block level operation types in fine grained updates

The Block level operation types in fine grained updates has six types: Whole block modification M – the whole blocks are replaced by a new data set; Partial modification PM – some part in a certain block is updated; Block deletion – all the blocks in the tree is deleted; Block insertion I – a new data is inserted in the tree and the whole block is created based on the tree structure; Block splitting SP – a part of the data in the block is taken out and a new block is inserted in the next.

(2) Setup

This method is similar to BLS based model except the file blocks are segment. The bilinear map is e: $G \times G \rightarrow G_T$ and G is GDH group. The collision resistant hash function is H: $(0,1)^* \rightarrow G$ and cryptographic hash function is h. After all parties finished the above fundamental parameters the following algorithm is run by the client.

Keygen(l^k): The secret value $\alpha \in Z_p$ is generated by the client and generate g from G, computing $v = g^{\alpha}$. The signing secret key pair {signpk, signsk} is selected by a designated provable secure model and the signing algorithm is sign(). The output of this algorithm is {signsk, α} as sk is secret key and { signpk,v,g} as pk is public key. The key pair signature is signpk = {v,g}, signsk = α.

FPP(F,sk,segreq): FPP denotes the file preprocess which includes segmentation requirement segreq which has a upper bound number s_{max} in each segment. Segment the file F to f = $\{m_{ij}\}$, where I ϵ (1,l), J ϵ (1,s), $s_i \in (1,s_{max})$. F is segmented as l blocks of total, with i^{th} block which has s_i segments. In the proposed method the size of each segment is taken as 40 bytes with 160-bit security. The client computes the HLA σ_i for every block. $\sigma_i = (H(m_i) \prod_{t \in I} v_i m_{ik}$ which produces the order set $\varphi = \{\sigma_i\}_{i \in (1,l)}$. the client generates root R which is based on RMHT T construction over $H(m_i)$. The signature is computed by $sig = (H(R))^{\sigma}$ and $u = (U_1 \| \ldots . \| U_{smax})$. The file tag for F is computed by client as $t = name \| n \| u \| Sig_{signsk}$ and forms the output {f,φ,T,R,sig,t}.

(3) Authorization Preparation

Every user in the cloud should authorize themselves to enter into the cloud server. The client request TPA for VID for security purposes. TPA returns its VID that is encrypted with client public key. The client computes $sig_{AUTH} = Sig_{signsk}(AUTH \| t \| VID)$ and sig_{AUTH} sends to TPA along with auditing delegation to compose the challenge later. After the algorithm is executed the client keeps RMHT with ranks in each node. This indicates there is a reduce in fine grained update request in each data block. The client sends {F,t,φ,sig,AUTH} to cloud server and the local server deletes { F,f,t,φ,sig}. the cloud server constructs RMHT T which is based on m_i and it keep T along with {F,t,φ,sig,AUTH} is stored for later verification.

(4) Data Updating Verification

The process is between the client and the cloud server. In this case there are five block level updates that affect T: PM, M, D, J and SP. In this section we discuss about the request in fine-grained update. The PM type of update is as follows and shown in figure 2.

(i) The client sends an update request UpdateReq to cloud server.

(ii) The cloud server performs the following operations:
PerformUpdate(UpdateReq, f) – Cloud server analyses UpdateReq to get $\{PM,I,o,m_{new}\}$ and the type is PM cloud server will update T and m_i and produces the output $P_{update} = \{m_i,\Omega_i,R',sig\}$ and the file f' is updated. After the completion of the algorithm P_{update} is send to the client by the cloud server.

(iii) The client will perform the following operations after receiving P_{update}.
VerifyUpdate(pk,P_{update}) – the client process m_i' by using $\{m_i$,UpdateReq) and parse P_{update} to $\{m_i,\Omega_i,R',sig\}$. the computes H(R) and R_{new} using { m_i,Ω_i} and {m_i', Ω_i}. the client verifies sig and H(R) and verifies if $R_{new} = R'$. If the condition fails the output is FALSE and return to cloud server, else the output is TRUE. If the output become TRUE the client performs the operation $\sigma_i' = (H(m_i) \prod_{t \in I} v_i m_{ik})^{\alpha}$ and $sig' = (H(R'))^{\alpha}$ and sends σ_i' and sig' to cloud sever.

(iv) The cloud server updates sig to sig' and σ_i to σ_i' respectively and deletes f if it receives {σ_i', sig'} the in indicates TRUE else it is FALSE and again run PerformUpdate(). The verification is repeatedly returning FALSE for the cheating request.

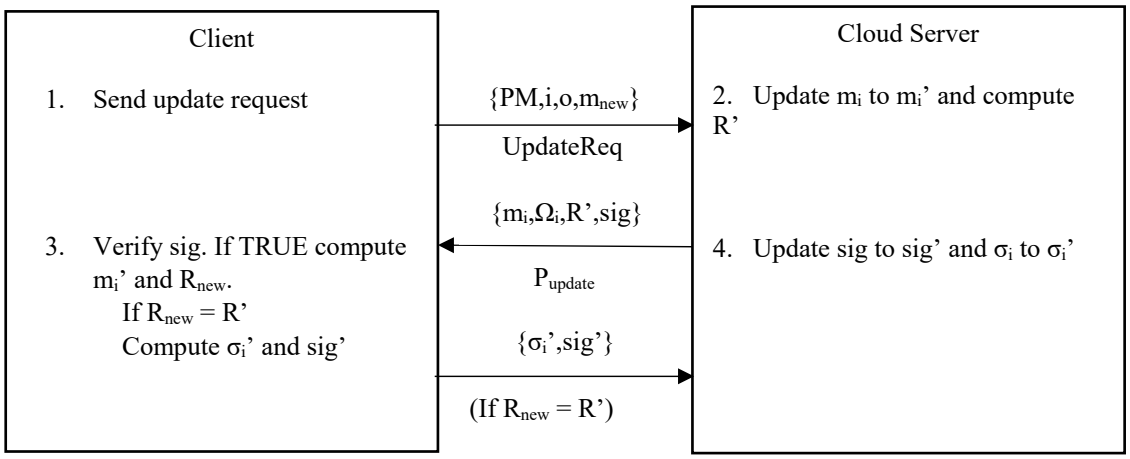

Figure 2: Verifiable Data Update

The other operations are performed as follows. The operations M, D, and I can compute σ_i' directly from the original file f without retrieving the data that is stored on cloud server. So, in the first phase the client sends σ_i' with UpdateReq. The cloud server sends $H(m_i)$ as response to client instead of m_i. For SP-type updating a new m* block is inserted in T after m_i' in addition with m_i to m_i'. the m* content is part of m_i. The cloud server sends m_i back to the client. Afterwards the process is similar to PM-type upgrade. In this case the client compute R_{new} by using $\{m_i', h(m^*), \Omega_i\}$ to compare R' instead of $\{m_i', \Omega_i\}$.

4.2. *Generation of challenge, proof and verification*

Before challenging a particular file, the TPA shows the cloud server that the file owner is authorized it and generation of Challenge, Proof and Verification algorithm is run by TPA and explained in figure 3.

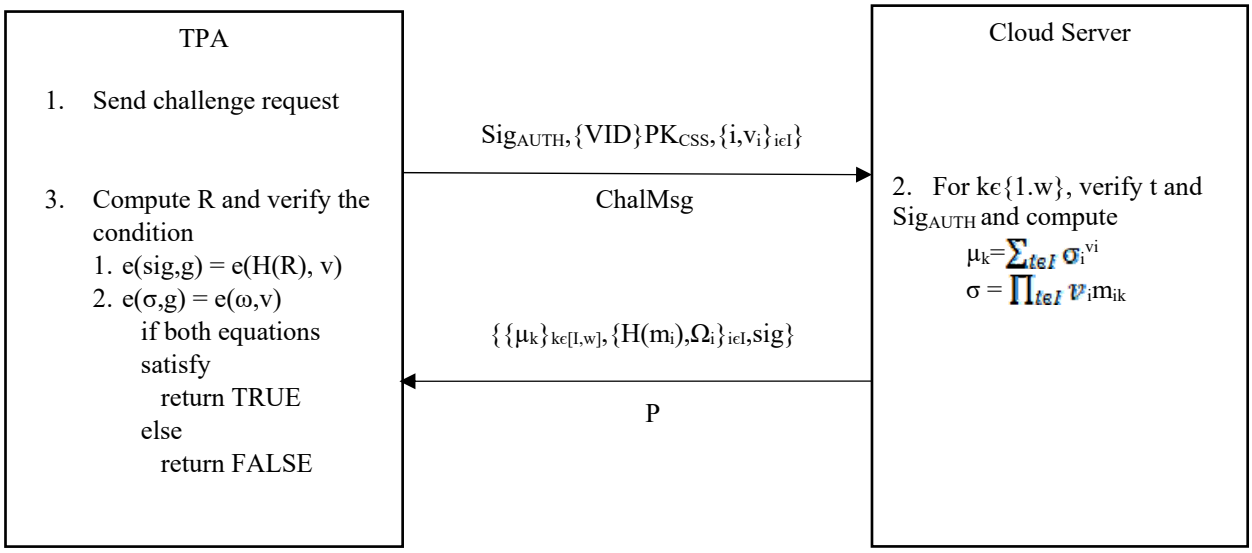

Figure 3: Challenge and Proof Generation and Verification

(1) ChallengeGen(Acc,pk,sig_{AUTH}) – The accuracy in the auditing is very much important. So, TPA will verify c from the total l blocks. The challenge message is generated as ChalMsg = {sig_{AUTH}, {VID}PK_{CS}, $(I,v_i)_{iϵl}$} and TPA's ID is VID. I is randomly selected from the subset [1,l] having c elements and $\{v_i ϵ Zp\}_{iϵl}$ are randomly chosen coefficient c. The VID is encrypted with the public key PK_{CS} and later decrypted with the secret key. TPA sends ChalMsg to cloud server.

(2) The cloud server receives the ChalMsg and runs the proof generation algorithm.
ProogGen(pk,F,sig_{AUTH},φ,ChalMsg) – Cloud sever verifies sig_{AUTH} with VID, t, AUTH and the client public key spk. If the output is false it rejects else cloud server computes compute $\mu_k = \sum_{tϵI} \sigma_i^{vi}$, kϵ[1,w], and $\sigma = \prod_{tϵI} v_i m_{ik}$. Consider if $k>s_i$ then $m_{ik}=0$ and proof P = {$\{\mu_k\}_{kϵ[1,w]}$, $\{H(m_i),\Omega_i\}_{iϵl}$, sig}. Then the cloud server sends p to TPA

(3) TPA receives p from cloud server and verification is done by the verification algorithm.
Verify(pk,ChalMsg, P) – The value of R is computed by TPA by using $\{H(m_i),\Omega_i\}$ and verifies sig using public keys v and g by comparing e(sig,g) and (H(R),v). if both are equal then it computes $\omega = \prod_{tϵI} v_i m_{ik}$ and it further verifies e(σ,g) and e(ω,v). If both are equal it returns TRUE else return FALSE.

5. Experimental Results and Evaluation

Hadoop [30] is installed that provides MapReduce programming and distributed file system. The implementation of this model is done in cloud environment using virtual machine with 32GB RAM, 1TB storage and 18 CPU cores. we use adult dataset and for each testing we use 1GB data. It has s number of sectors and 20 MB memory is allocated for each sector. In the initial uploading and splitting we use $s_i = s_{max}$ and s_{max} denotes the maximum number of blocks. First, we test how s_{max} influence the proof size P. Figure 4 shows the proof size generally decreases then the size of s_{max} decreases when f is initially uploaded. The storage of RMHT at cloud server will decrease when there is an increase in average number of blocks. So, a large s_{max} is recommended in dynamic settings.

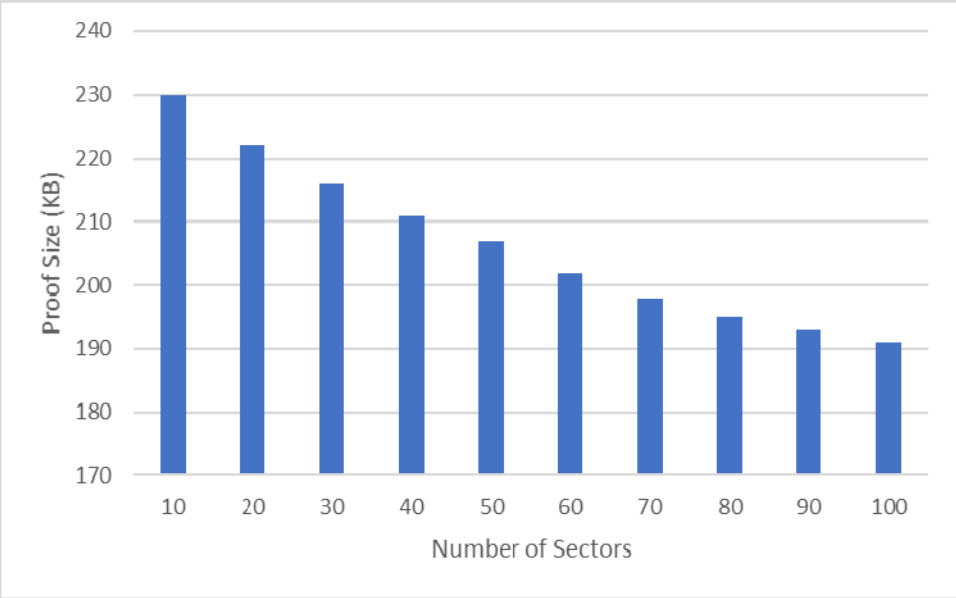

Figure 4: Communication Overhead for different s_{max}

Second, storage overhead for inserting a small block is tested. Without fine grained updates each small insertion creates a new block and leads to waste of storage memory. We compare the proposed model with public auditing model. The updates taken for the experiments are 10 * 280 bytes which is filled with the random data. The results are shown in figure 5. The updates use the same size of storage in the cloud server becomes constant and in the proposed method and the storage capacity is linearly increase. The result shows that the proposed model with fine grained data update incur lower storage for small insertion when compare with the existing model.

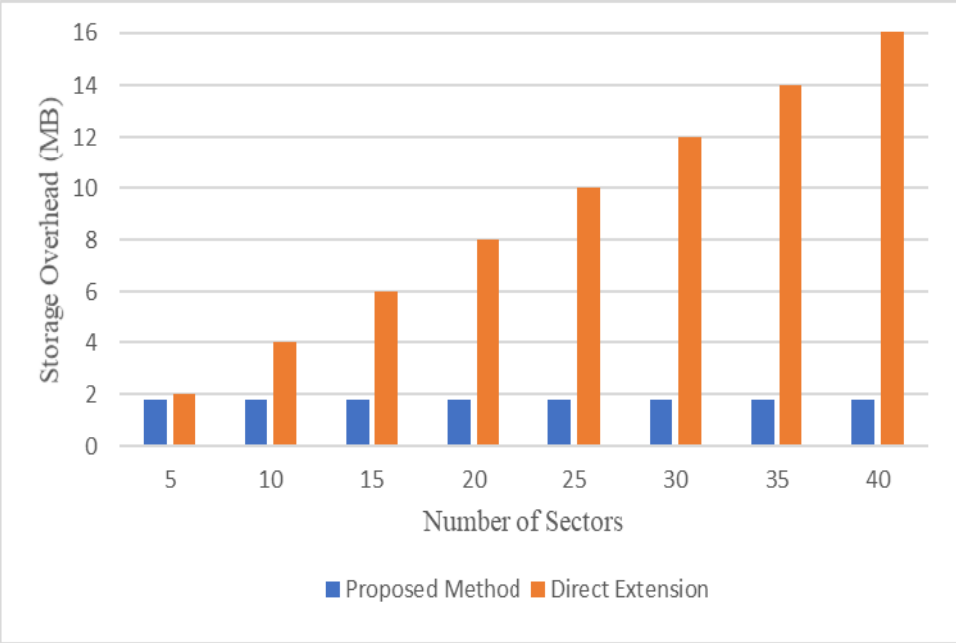

Figure 5: Comparison of total storage overhead inserted different number of sectors between proposed method and Direct Extension

Third, the performance analysis during the modification is analyzed. Here three various blocks of data with size 100 bytes, 140 bytes and 180 bytes are taken. Each block contains 20 to 40 standards with 20 bytes sector. For verifiable phase updating the key factor is the data retrieval in communication overhead. For every update, the total amount for data retrieval is recorded for proposed method and existing method is shown in figure 6. We see that the efficiency of the proposed method is better in data retrieval communication overheads and has the advantage for more significant larger updates. When update is same size the advantage gets decreased. when there is an increase in $|s_i|$ large number of sectors in original file will be retrieved. So, the size of the block should be kept low in case of less communication verifiable updates.

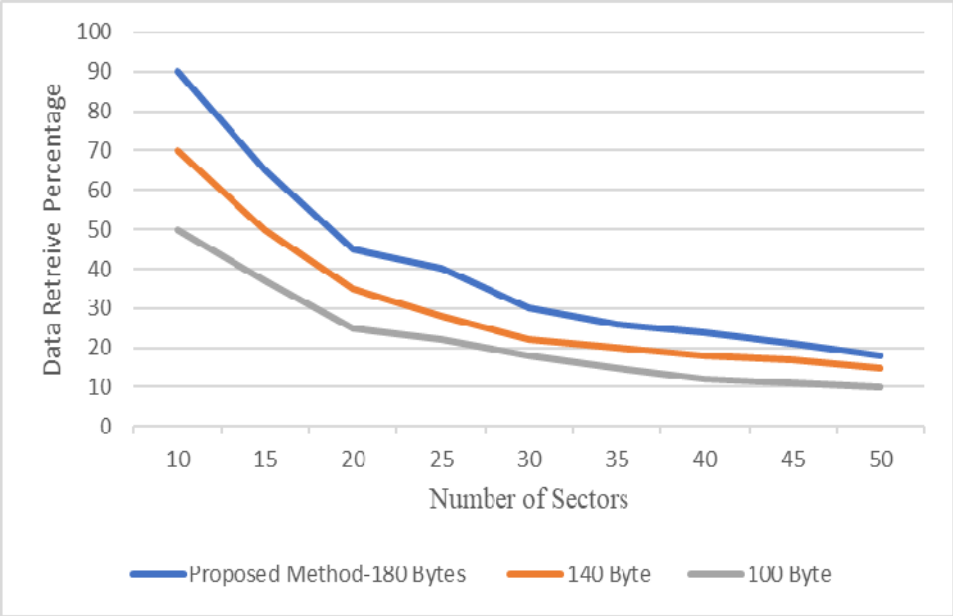

Figure 6: Percentage of saving in communication overhead during data retrieval in the proposed method and existing method.

From the result in small updates we see that the proposed method incurs low storage overhead when compared to the existing method which will reduce the communication overhead when compare with existing method. The parameter s_{max} should be chosen carefully depending on variable data size and efficiency in communication and storage. For example, for application with similar scale of 1GB data and 160-byte updates and $s_{max} = 30$ allows the method incur lower overhead in both communication and storage during updates.

6. Conclusion

This paper provides formal analysis in different fine-grained data updates. The proposed method supports fine-grained update and authorized auditing and also this method reduces the communication overheads when verifies small updates. The experiment result demonstrate that our method offers flexibility and security and also has lower overheads for small updates which is frequently happen in business transactions and social media and is mainly useful for big data applications. So, this paper improves data auditing in big data.

References

[1] H.Abbas, O.Maennel, and S.Assar, ‘‘Security and privacy issues in cloud computing,’’ Ann. Tele commun., vol. 72, pp. 5–6, pp. 233–235, 2017.

[2] M. Sookhak, A. Gani, M. K. Khan, and R. Buyya, ‘‘Dynamic remote data auditing for securing big data storage in cloud computing,’’ Inf. Sci., vol. 380, pp. 101–116, Feb. 2017.

[3] T. S. Fun, A. Samsudin, and Z. F. Zaaba, ‘‘Enhanced security for public cloud storage with honey encryption,’’ Adv. Sci. Lett., vol. 23, no. 5, pp. 4232–4235, 2017.

[4] D. A. B. Fernandes, L. F. B. Soares, J. V. Gomes, M. M. Freire, and P. R. M. Inácio ‘‘Security issues in cloud environments: A survey,’’ Int. J. Inf. Secur., vol. 13, no. 2, pp. 113–170, Apr. 2014.

[5] M. N. Kulkarni, B. A. Tidke, and R. Arya, ‘‘An improved privacy preserving public auditing for secure cloud storage,’’ in Proc.5thInt.Conf. Soft Comput. Problem Solving. Singapore: Springer, 2016, pp. 853–866.

[6] V.Chang, Y.-H.Kuo, and M.Ramachandran,‘‘Cloud computing adoption framework: A security framework for business clouds,’’ Future Gener. Comput. Syst., vol. 57, pp. 24–41, Apr. 2016.

[7] Customer Presentations on Amazon Summit Australia, Sydney, 2012, accessed on: March 25, 2013. [Online]. Available: http://aws.amazon.com/apac/awssummit-au/.

[8] J. Yao, S. Chen, S. Nepal, D. Levy, and J. Zic, ‘‘TrustStore: Making Amazon S3 Trustworthy With Services Composition,’’ in Proc. 10th IEEE/ACM Int’l Symposium on Cluster, Cloud and Grid Computing (CCGRID), 2010, pp. 600-605.

[9] D. Zissis and D. Lekkas, ‘‘Addressing Cloud Computing Security Issues,’’ Future Gen. Comput. Syst., vol. 28, no. 3, pp. 583-592, Mar. 2011.

[10] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, ‘‘Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing,’’ IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 5, pp. 847-859, May 2011.

[11] C. Wang, Q. Wang, K. Ren, and W. Lou, ‘‘Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing,’’ in Proc. 30st IEEE Conf. on Comput. and Commun. (INFOCOM), 2010, pp. 1-9.

[12] G. Ateniese, R.D. Pietro, L.V. Mancini, and G. Tsudik, ‘‘Scalable and Efficient Provable Data Possession,’’ in Proc. 4th Int’l Conf. Security and Privacy in Commun. Netw. (SecureComm), 2008, pp. 1-10.

[13] G. Ateniese, R. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. Peterson, and D. Song, ‘‘Remote Data Checking Using Provable Data Possession,’’ ACM Trans. Inf. Syst. Security, vol. 14, no. 1, May 2011, Article 12.

[14] G.Ateniese, R.B.Johns, R.Curtmola, J.Herring, L.Kissner, Z.Peterson, and D. Song, ‘‘Provable Data Possession at Untrusted Stores,’’ in Proc. 14th ACM Conf. on Comput. and Commun. Security (CCS), 2007, pp. 598-609.

[15] R. Curtmola, O. Khan, R.C. Burns, and G. Ateniese, ‘‘MR-PDP: Multiple-Replica Provable Data Possession,’’ in Proc. 28th IEEE Conf. on Distrib. Comput. Syst. (ICDCS), 2008, pp. 411-420.

[16] C. Erway, A. Ku ¨pc¸u ¨, C. Papamanthou, and R. Tamassia, ‘‘Dynamic Provable Data Possession,’’ in Proc. 16th ACM Conf. on Comput. and Commun. Security (CCS), 2009, pp. 213-222.
[17] S. Nepal, S. Chen, J. Yao, and D. Thilakanathan, ‘‘DIaaS: Data Integrity as a Service in the Cloud,’’ in Proc. 4th Int’l Conf. on Cloud Computing (IEEE CLOUD), 2011, pp. 308-315.
[18] Y. He, S. Barman, and J.F. Naughton, ‘‘Preventing Equivalence Attacks in Updated, Anonymized Data,’’ in Proc. 27th IEEE Int’l Conf. on Data Engineering (ICDE), 2011, pp. 529-540.
[19] E. Naone, ‘‘What Twitter Learns From All Those Tweets,’’ in Technology Review, Sept. 2010, accessed on: March 25, 2013. [Online]. Available: http://www.technologyreview.com/ view/420968/what-twitter-learns-from-all-those-tweets/
[20] X. Zhang, L.T. Yang, C. Liu, and J. Chen, ‘‘A Scalable Two-Phase Top-Down Specialization Approach for Data Anonymization Using MapReduce on Cloud,’’ IEEE Trans. Parallel Distrib. Syst., vol. 25, no. 2, pp. 363-373, Feb. 2014.
[21] S.E. Schmidt, ‘‘Security and Privacy in the AWS Cloud,’’ presented at the Presentation Amazon Summit Australia, Sydney,Australia, May2012, accessedon: March 25, 2013. [Online]. Available: http://aws.amazon.com/apac/awssummit-au/.
[22] C. Liu, X. Zhang, C. Yang, and J. Chen, ‘‘CCBKEVSession Key Negotiation for Fast and Secure Scheduling of Scientific Applications in Cloud Computing,’’ Future Gen. Comput. Syst., vol. 29, no. 5, pp. 1300-1308, July 2013.
[23] X. Zhang, C. Liu, S. Nepal, S. Panley, and J. Chen, ‘‘A Privacy Leakage Upper-Bound Constraint Based Approach for CostEffective Privacy Preserving of Intermediate Datasets in Cloud,’’ IEEE Trans. Parallel Distrib. Syst., vol. 24, no. 6, pp. 1192-1202, June 2013.
[24] A. Juels and B.S. Kaliski Jr., ‘‘PORs: Proofs of Retrievability for Large Files,’’ in Proc. 14th ACM Conf. on Comput. and Commun. Security (CCS), 2007, pp. 584-597.
[25] H. Shacham and B. Waters, ‘‘Compact Proofs of Retrievability,’’ in Proc. 14th Int’l Conf. on Theory and Appl. of Cryptol. and Inf. Security (ASIACRYPT), 2008, pp. 90-107.
[26] D. Boneh, H. Shacham, and B. Lynn, ‘‘Short Signatures From the Weil Pairing,’’ J. Cryptol., vol. 17, no. 4, pp. 297-319, Sept. 2004.
[27] G. Ateniese, S. Kamara, and J. Katz, ‘‘Proofs of Storage From Homomorphic Identification Protocols,’’ in Proc. 15th Int’l Conf. on Theory and Appl. of Cryptol. and Inf. Security (ASIACRYPT), 2009, pp. 319-333.
[28] Y. Zhu, H. Hu, G.-J. Ahn, and M. Yu, ‘‘Cooperative Provable Data Possession for Integrity Verification in Multi-Cloud Storage,’’ IEEE Trans. Parallel Distrib. Syst., vol. 23, no. 12, pp. 2231-2244, Dec. 2012.
[29] J.L. Joneston Dhas, S. Maria Celestin Vigila and C. Ezhil Star, “A Framework on Security and Privacy Preserving for Storage of Health Information Using Big Data,” International Journal of Control Theory and Applications, 10(03), pp.91-100, 2017, International Science Press.
[30] Hadoop MapReduce. [Online]. Available: http://hadoop. apache.org.